A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks

Conference Paper · December 2018

DOI: 10.1109/CTEMS.2018.8769211

Eustace Dogo
University of Johannesburg South Africa

Oluwatobi Afolabi
University of Johannesburg

Nnamdi Nwulu
University of Johannesburg

Bhekisipho Twala
Durban University of Technology

A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks

E. M. Dogo[1], O. J. Afolabi[2], N. I. Nwulu[3], B. Twala[4], C.O. Aigbavboa[5]
[1,2,3]Department of Electrical and Electronics Engineering Science, [5]Department of Construction Management and Quantity Survey
University of Johannesburg, Johannesburg, South Africa
[4]School of Engineering, Department of Electrical and Mining Engineering, University of South Africa
{[1]eustaced, [2]afolabij, [3]nnwulu, [5]caigbavboa}@uj.ac.za, [4]twalab@unisa.ac.za

***Abstract*—In this paper, we perform a comparative evaluation of seven most commonly used first-order stochastic gradient-based optimization techniques in a simple Convolutional Neural Network (ConvNet) architectural setup. The investigated techniques are the Stochastic Gradient Descent (SGD), with vanilla (vSGD), with momentum (SGDm), with momentum and nesterov (SGDm+n)), Root Mean Square Propagation (RMSProp), Adaptive Moment Estimation (Adam), Adaptive Gradient (AdaGrad), Adaptive Delta (AdaDelta), Adaptive moment estimation Extension based on infinity norm (Adamax) and Nesterov-accelerated Adaptive Moment Estimation (Nadam). We trained the model and evaluated the optimization techniques in terms of convergence speed, accuracy and loss function using three randomly selected publicly available image classification datasets. The overall experimental results obtained show Nadam achieved better performance across the three datasets in comparison to the other optimization techniques, while AdaDelta performed the worst.**

***Keywords*— Artificial Intelligence, optimizers, performance measures, deep learning, stochastic gradient descent**

I. INTRODUCTION

Convolutional Neural Networks (ConvNet) is one the of the state-of-art deep learning technique and architecture that has over the past few years produced remarkable results in computer vision problem domain, like in the ImageNet Large Scaled Visual Recognition Challenge (ILSVRC) project which has witnessed a steady reduction in error rate between 2010 and 2017. In 2017 the winning algorithm achieving a remarkable classification error rate of 2.3%, surpassing the human classification error rate of 5% [1, 2].

Training of deep neural networks like ConvNet is to a large extent a challenging non-convex and high dimensional optimization problem. Depending on the domain problem and datasets, the goal of a deep learning researcher is to implement a model that will produce better and faster results through hyperparameters tuning to minimize the loss function. Optimally tuning the weights in deep neural networks is key to producing accurate model classification or prediction. However, tuning and adjusting the weights is dependent on setting weights with lowest loss function (gradient descent) and the direction of change of the slope during gradient descent (backpropagation). To avoid the problem of local minima trapping, as against obtaining the desirable absolute global minimum value, as well as other challenges associated with gradient descent such as curse of dimensionality, many optimization algorithms have been proposed in recent years, the trend mostly based on adaptive estimations that works automatically and allows for little tuning of the hyperparameters.

Non-convex problems are the most common case for neural networks, hence choosing optimization strategy that seeks to find the global optima in these networks is usually challenging due to process of estimation of a very large number of parameters in high dimensional search space. As an improper optimization technique may make the network to reside in the local minima during training without any improvement. Additionally, some neural network researchers intuitively show preference to Adam optimizer for all cases, due to its robustness and performance across numerous case scenarios. Hence, an investigation is required to analyse the performance of optimizers depending on the model and dataset employed for a better understanding of their behaviour. Thus, the contribution of this paper is an experimental comparison of the performance of seven (7) well known and commonly used first-order stochastic gradient descent optimization algorithms on a ConvNet model using three (3) different image classification datasets and reveal how well, fast and stable each optimizer was able to deal with the problem of finding the appropriate and optimal minima during training, using convergence speed, accuracy and loss function as performance criteria.

The rest of the paper is organized in the following order: Section 2 presents a background of key concepts and related works, followed by a brief description of the optimization techniques to be examined. The experimentation results and discussion are presented in section 3. Finally, section 4 concludes the paper.

II. BACKGROUND AND LITERATURE REVIEW

A. Overview of Neural Networks Optimization

Over the years, research on the improvement and development of new optimization algorithms has played a vital role in the improvement of deep learning architectures. Neural Networks to a very large extent is an optimization problem which seeks to find the global optimum through a robust training trajectory and fast convergence using gradient descent algorithms [3]. Optimization problem in data fitting is finding a model parameter values that are consistent with prior information and gives the best fit with the smallest prediction error with observed data. Optimization problem is mathematically defined as [4]:

978-1-5386-7709-4/18/$31.00 ©2018 IEEE

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$, Find $\hat{\boldsymbol{x}} = \text{argmin}\, f\,(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}$,

where *f* is called the objective function or cost function and $\hat{\boldsymbol{x}}$ is the minimizer of the objective function *f*.

B. Gradient Descent Variants

Gradient descent is a way to minimize an objective function $f\,(\boldsymbol{x})$ parameterized by a model's parameters $\boldsymbol{x} \in \mathbb{R}$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_x f(x)$ w.r.t. to the parameters, reaching the local minimum is determine by the learning rate [3]. Generally, there are three variants of gradient descent: 1) Batch Gradient Descent (BGD) – uses the entire dataset rows for training at the same time and then makes adjustments to the weights, and is hence referred to as a deterministic approach, 2) Stochastic Gradient Descent (SGD) – uses single training dataset at a time (one row after another) and then iteration adjustment of weights for each row and, 3) a hybrid of BGD and SGD which is called Mini-batch gradient descent that uses more than one training example at a time [5]. They differ based on the amount of data utilised to compute gradient of the objective function, usually making trade-off between accuracy of the parameter update and the time it takes to perform the update [3]. BGD works well in convex scenarios where finding a local minimum is an assurance of reaching a global minimum, however highly non-convex scenarios are common for neural networks and the challenge is avoiding getting trapped in suboptimal local minima. For this and other reasons such as choosing and adjusting learning rates, over the years gradient descent optimization algorithms have been developed for deep learning to mitigate challenges associated with gradient descent.

Gradient descent optimization algorithm is a well-studied area of deep learning research. Over the years, different optimization algorithms have been developed by researchers to improve on vanilla SGD to enhance the performance of deep convolutional neural networks. Optimization is one of the key components in deep learning, which helps a model to train better during backpropagation when the weights are adjusted to minimize loss error, as well as to address the curse of dimensionality problem [5]. The most popular optimization algorithm used in deep learning libraries such as in Caffe, Lasagne and Keras are: SGD [6], RMSProp [7], AdaGrad [8], AdaDelta [9], Adam [10, 11], Adamax [10] and Nadam [12]. Researchers continue to develop optimizers to achieve better generalization such as in [13].

CNN is the most widely used deep learning model in the areas of computer vision, image processing, speech recognition and natural language processing (NLP). CNN is a specific kind of neural network normally used for processing grid-like topology data, such as 1D grid time series data at regular intervals and 2D grid of pixels image data [5]. CNN employs a convolutional mathematical operation at the CNN layer to derive weighting function *w(a)*, where *a* is the age of measurement for a time series data. By applying a weighted average operation at every time interval, the convolution operation is generally defined as follows:

$$s(t) = \int x(a)w(t-a)da = s(t) = (x * w)(t) \qquad (1)$$

Where x is input and w as the kernel or filter, and s the output called the feature map for continues time series t. The discrete convolution operation with the assumption that x and w are defined based on integer values of t will assumes the following form:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \qquad (2)$$

For a 2-dimesional image *I* input, and 2-dimentional kernel *K*, the cross-correlation convolution is described as follows:

$$S(i,j) = (I * K)(i.j) = \sum_{m}\sum_{n} I(i+m, j+n)K(m,n) \quad (3)$$

CNN differs from the usual neural network layers by having 1) sparse interactions whereby the kernel is made smaller than the input, by using only meaningful features which translates to fewer parameters and operations to compute the output with better model statistical efficiency. 2) parameter sharing refers to the use of one weight value or parameter for multiple functions in a model, and 3) equivariant representation where output changes when input change in same manner [5].

A comparison of five different stochastic gradient descent based optimization methods namely: SGD and SGD with momentum, Adam, AdaGrad, AdaDelta and RMSProp were investigated by authors in [14], based on convergence time, number fluctuations and parameter update rate, using different number of iterations and specific test function values on a Stacked denoising autoencoder (SdA) architecture. Based on their experimental findings, AdaDelta had a superior performance compared to the other optimizers in terms of fast convergence. However, it is unclear what dataset was used for their experiment. In [15], authors carried out a performance comparison of four gradient descent-based variants namely, gradient descent, stochastic gradient descent, semi-stochastic gradient descent and stochastic average descent on logistic and softmax regression on synthetic and MNIST handwritten digits dataset respectively, limited convex objective fitting problem. Stochastic gradient descent generally performing better than SG in the two experiments conducted by the authors; whereas the two hybrid variants produced better accuracy within reasonable time. Similarly, a comparative performance study of gradient descent (GD) and SGD was conducted in [16] using performance metric as accuracy, training and convergence time on MNIST dataset for linear regression and multinomial logistic regression. A recent study by [3], gave an intuitive overview of the behaviour of modern gradient descent optimization algorithms. The author's intuitive opinion was to use adaptive learning rate methods if training deep and complex neural network models and faster convergence. Also highlighted were the challenges associated with these optimizers with strategies that could be utilised to improve optimizing gradient descent. The authors in [17], through practical experimentation, demonstrated sources of gradient descent performance failures in deep learning, they attributed these failures to subtle problems such as informativeness of the gradient descents, signal to noise ratio (SNR), bad conditioning associated with activations which leads to vanishing gradient and slow training process. From these related works it seems that an extensive study to compare the performance effects of these popular and well used optimization algorithms on a convolutional neural network architecture using image classification datasets has not been entirely conducted. Most studies have investigated GD versus SGD or a couple of these optimizers against certain metrics and models. Others have intuitively shed light on tricks and strategies to improve the optimization algorithms and common problems associated with these optimizers. Ours is a practical experimentation to find out the impact of these seven well utilised optimizers using three standard image classification datasets on a simplified convolutional architecture.

III. EXPERIMENTAL SETUP AND RESULTS

We carried out our experiment on three image classification dataset problems using a ConvNet model. We showed the effectiveness of seven (7) popular stochastic gradient based optimization algorithms used in deep learning realm on the built model. The optimizers we compared were SGD (vanilla, with momentum, and nesterov), RMSProp, Adam, Adamax, Adagrad, Adadelta, and Nadam, while the dataset used were: 'Cats and Dogs', 'Fashion MNIST' and 'Natural Images' obtained from Kaggle's public dataset platform. Generally, we applied all the optimization algorithms on each dataset using our ConvNet model. We maintained the same hyperparameters settings for all experiments. Dropout was added to the output layer of the ConvNet model to address overfitting and the whole network was trained over 450 epochs for the 'Fashion MNIST' and the 'Cats and Dogs' dataset but was trained over 150 epochs for the 'Natural Images' dataset. The Natural Images dataset was trained over 150 epochs so that the average number of training hours (which is 5 hours) is kept constant for all experiments. This does not have any negative impact on the Natural Images models since convergence was already approached at about 140 epochs for most of the optimizers.

A. *ConvNet Model*

The ConvNet architecture employed in this study has a 3x3 sized filters across all convolution layers and 3x3 Maxpooling with stride size equal to 1, followed by a fully connected layer. The model was built with the keras deep learning library and trained using Kaggle's NVidia K80 with 12GB VRAM GPU for about 5 hours per optimizer. Dropout and data augmentation were two regularization techniques which were applied during the training of the model. Rectifier Linear Unit (ReLU) nonlinearity activation function was also applied to each convolutional layer. A summary of the ConvNet configuration applied in this study is outlined in Table 1.

TABLE 1. SUMMARY OF CONVNET CONFIGURATIONS

Dataset-1: Cats and Dogs	**Dataset-2: Fashion MNIST**	**Dataset-3: Natural Images**
Input image data - 64x64x3-channels RGB Images	Input data image - 28x28x1-channel grayscale Images	Input date image - 150x150x3-channels RGB Images
Conv3x3-32; stride=1	Conv3x3-32; stride=1	Conv3x3-32; stride=1
ReLU (nonlinearity function)		
Pooling layer: MaxPooling 2x2; stride= 1		
Conv3x3-32; stride=1	Conv3x3-32; stride=1	Conv3x3-32; stride=1
ReLU (nonlinearity function)		
Pooling layer: MaxPooling 2x2; stride= 1		
Conv3x3-64; stride=1	Conv3x3-64; stride=1	Conv3x3-64; stride=1
ReLU (nonlinearity function)		
Pooling layer: MaxPooling 2x2; stride= 1		
Flattening (Input Layer of Neural Network)		
FC-64 layers		
ReLU (nonlinearity function)		
Dropout = 0.5 (addressing overfitting issue)		
Binary Cross entropy	Categorical Cross entropy	Categorical Cross entropy
Sigmoid Layer (ϕ)	Softmax Layer (σ)	Softmax Layer (σ)

B. *Datasets*

In this work, we choose three (3) popular image classification problem datasets to evaluate our model with the different optimization algorithms.

1) *Performance Assessment of Dataset-1 (Cats and Dogs):* This dataset was provided by Microsoft Research and referred to as Asirra dataset, it is made up input size of 64x64 pixels of dogs and cats coloured images. The training set contains 25,000 images, comprising 12,500 images of dogs and 12,500 images of cats, with a test dataset of 12,500 images [18]. The simulation results are shown in Table 2 and on Figures 1-9.

2) *Performance Assessment of Dataset-2 (Fashion MNIST):* Fashion-MNIST is a dataset of Zalando's article images, consisting of a training set of 60,000 image examples and a test set of 10,000 image examples. Each example is a 28x28 grayscale images, associated with a label from 10 classes, with 7,000 images per class. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits [19]. The simulation results are shown in Table 3 and on Figures 10-18.

3) *Performance Assessment Dataset-3 (Natural Images):* This dataset is made up of 6,899 images of 150x150 pixels coloured images from 8 distinct classes, which include airplane, car, cat, dog, flower, fruit, motorbike and person obtained from various sources, and is used as a benchmark dataset for the work in [20]. The simulation results are shown in Table 4 and on Figures 19-27.

TABLE 2. RESULTS FOR CATS & DOGS DATASET

Optimizers	Convergence time (hrs)	Accuracy	Loss
vSGD	2.625	0.643	0.670
SGDm	2.375	0.745	0.525
SGDm+n	2.250	0.736	0.531
AdaGrad	2.500	0.675	0.649
RMSProp	2.125	0.848	0.429
Adam	2.625	0.829	0.400
AdaDelta	2.625	0.624	0.687
Adamax	2.250	0.826	0.437
Nadam	**2.625**	**0.855**	**0.362**

TABLE 3. RESULTS FOR FASHION MNIST DATASET

Optimizer	Convergence time (hrs)	Accuracy	Loss
vSGD	2.500	0.373	1.886
SGDm	2.375	0.585	1.278
SGDm+n	2.625	0.574	1.317
AdaGrad	2.625	0.404	2.048
RMSProp	2.5	0.458	1.648
Adam	**2.625**	**0.720**	**0.920**
AdaDelta	2.750	0.272	2.220
Adamax	2.250	0.622	1.189
Nadam	2.625	0.712	0.941

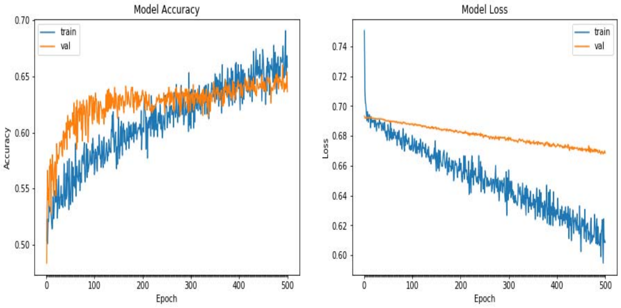

Fig. 1. SGD vanilla

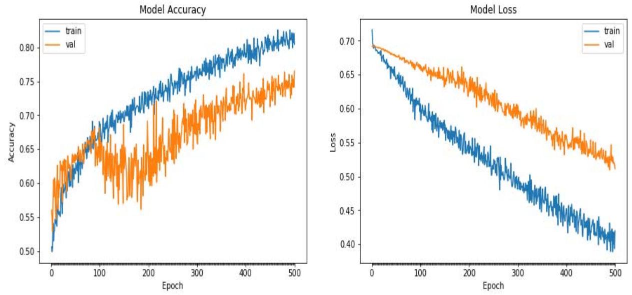

Fig. 2. SGD momentum

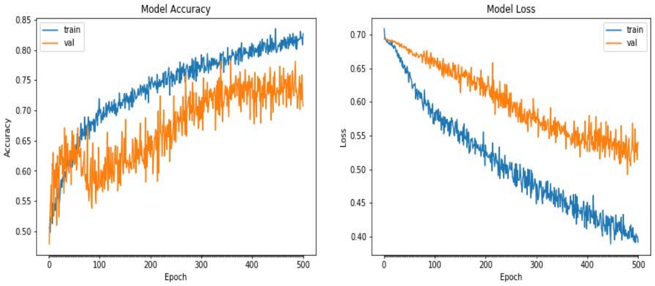

Fig. 3. SGD momentum + nesterov

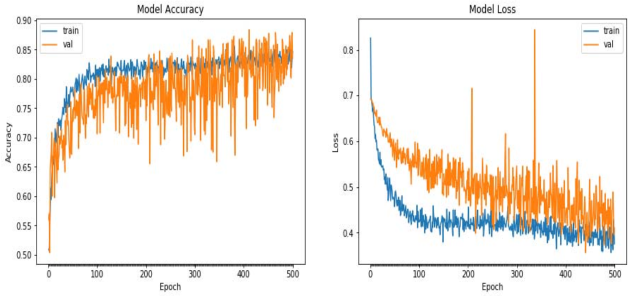

Fig. 4. RMSProp

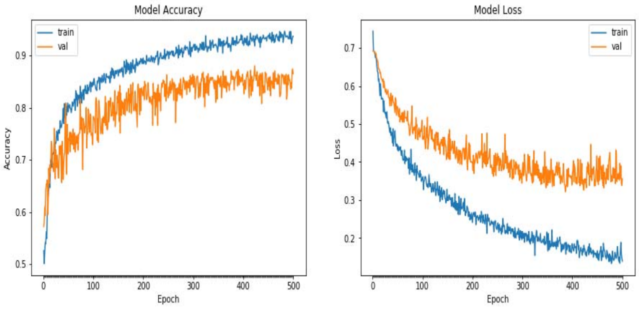

Fig. 5. Nadam

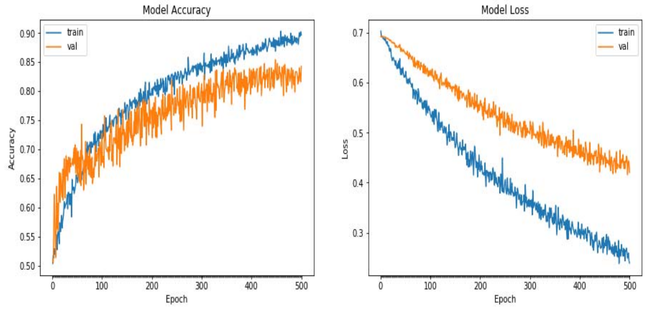

Fig. 6. Adamax

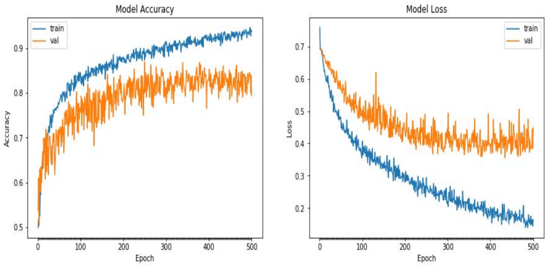

Fig. 7. Adam

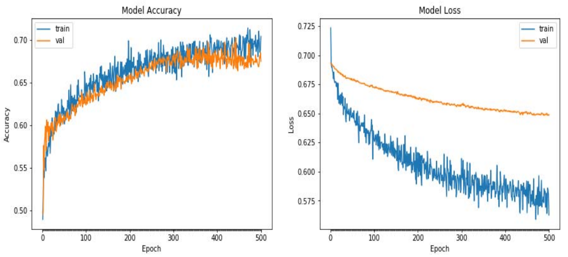

Fig. 8. AdaGrad

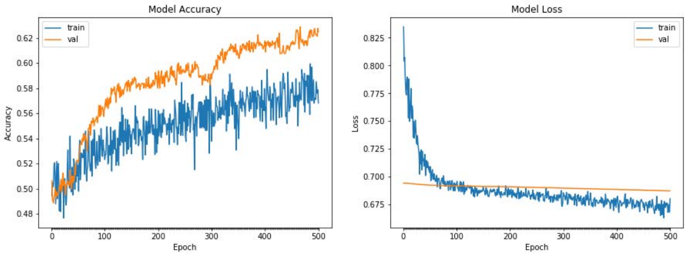

Fig. 9. AdaDelta

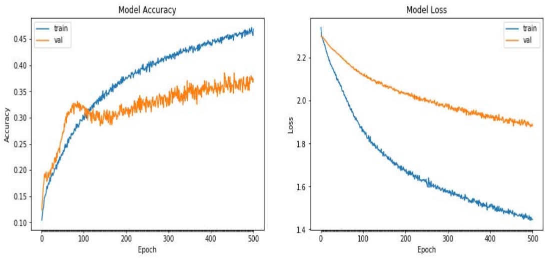

Fig. 10. SGD vanilla

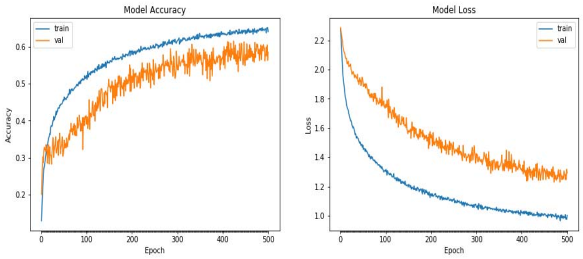

Fig. 11. SGD momentum

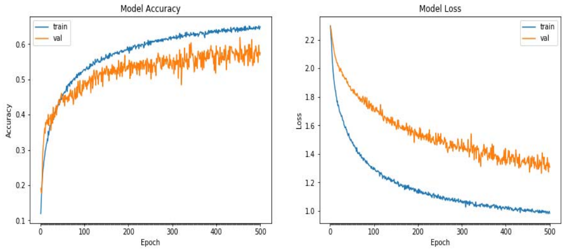

Fig. 12. SGD momentum + nesterov

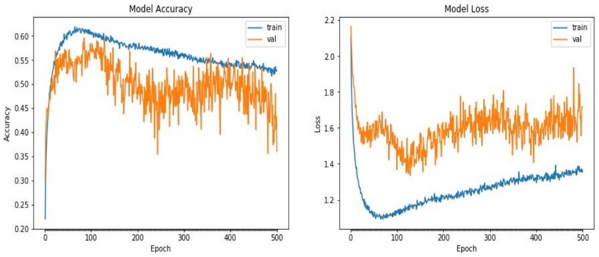

Fig. 13. RMSProp

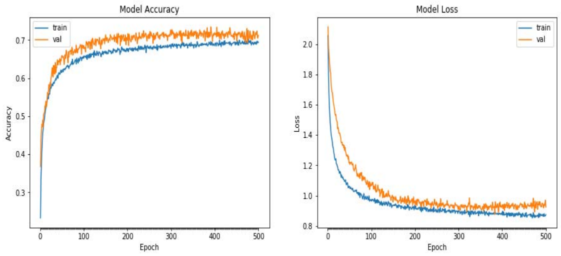

Fig. 14. Nadam

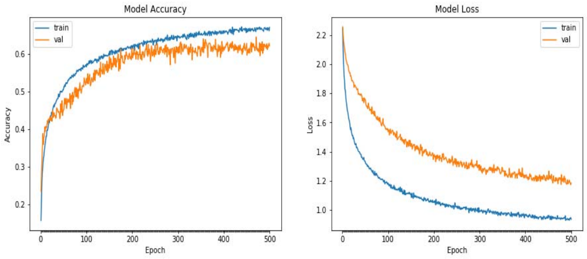

Fig. 15. Adamax

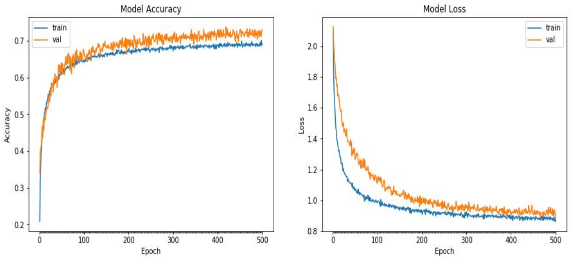

Fig. 16. Adam

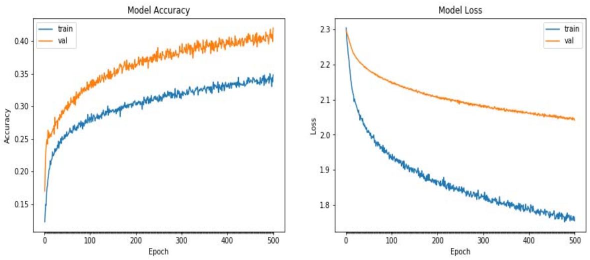

Figure 17: AdaGrad

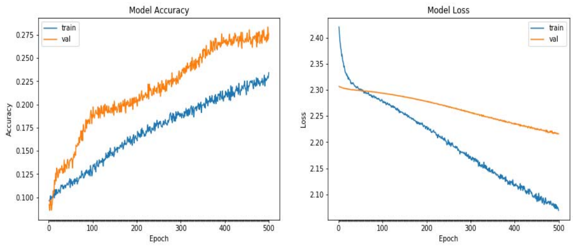

Figure 18: AdaDelta

TABLE 4. RESULTS FOR NATURAL IMAGES DATASET

Optimizer	Convergence Time (hrs)	Accuracy	Loss
vSGD	5.056	0.775	1.228
SGDm	4.861	0.891	0.501
SGDm+n	5.017	0.851	0.584
AdaGrad	5.017	0.699	1.589
RMSProp	5.172	0.797	0.619
Adam	5.056	0.892	0.920
AdaDelta	5.483	0.739	1.669
Adamax	4.900	0.909	0.440
Nadam	5.444	**0.913**	**0.225**

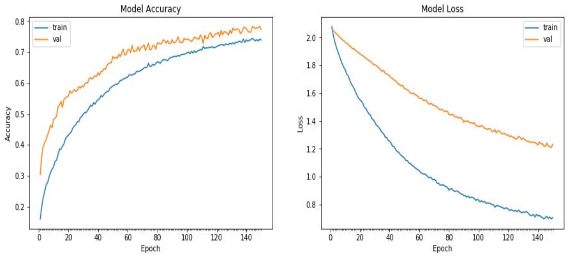

Fig. 19. SGD vanilla

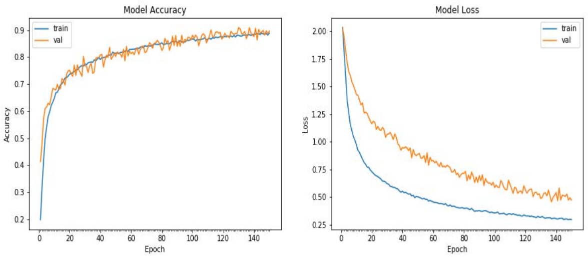

Fig. 20. SGD momentum

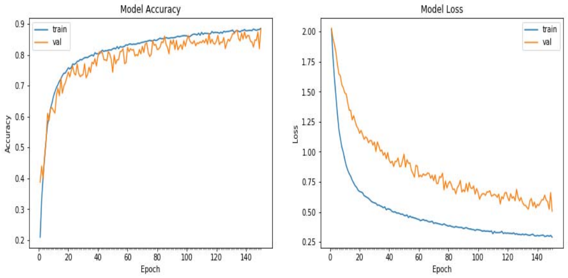

Fig. 21. SGD momentum + nesterov

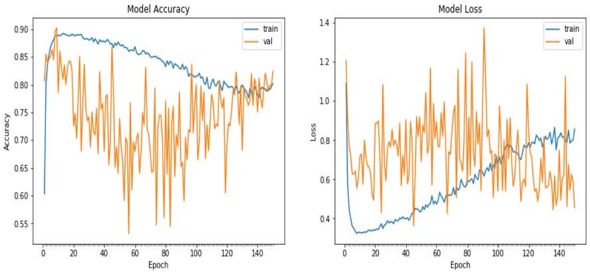

Figure 22: RMSProp

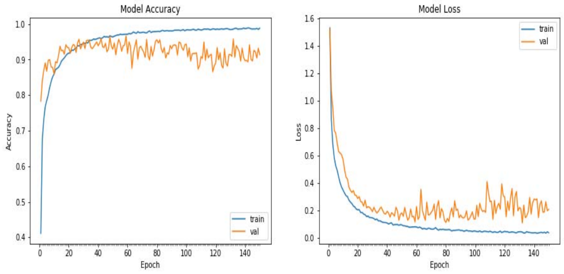

Fig. 23. Nadam

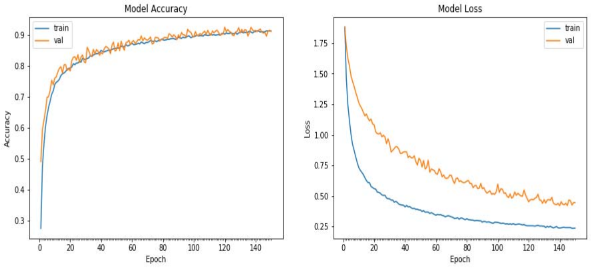

Fig. 24. Adamax

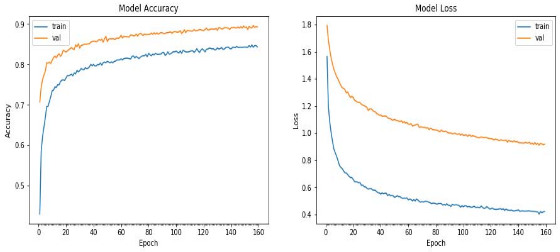

Fig. 25. Adam

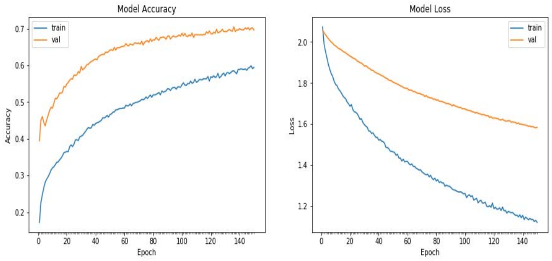

Fig. 26. AdaGrad

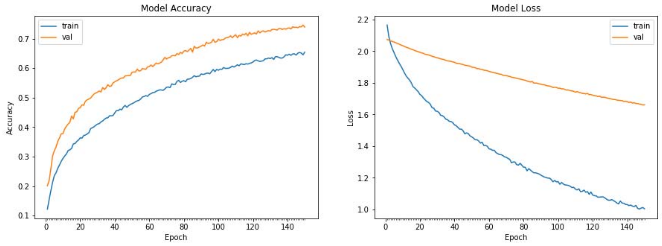

Fig. 27. AdaDelta

C. Overall Results Discussion

1) Dataset 1 – Cats and Dogs: Table 2 and Figures 1-9 present the results for the Cats and Dogs dataset image classification problem, which shows Nadam outperforming the optimizers, but closely trailed by RMSProp and with no significant difference between Adam (third) and Adamax (fourth). The convergence was at 450 epochs and an average of 2.5 hours convergence time for all the optimizers. AdaDelta, vSGD and AdaGrad were the least performers on this dataset.

2) Dataset 2 – Fashion MNIST: Generally, all the optimizers did not perform well on the Fashion MNIST dataset as presented on Table 3 and Figures 10-18, which could be because of the simplified ConvNet architecture utilised and size of the dataset. However, Adam was the better performer on this dataset with an accuracy of 72%, marginally better than Nadam that achieved an accuracy of 71.2%. The rest optimizers performed woefully especially AdaDelta with an accuracy of just 27.2%. RMSProp not only underperformed for this dataset but was unstable throughout training. RSMProp could not converge during training, it was observed that reading was taken for the first 450 epoch during which other optimizers had converged. It is evident that all the optimizers got a good test on this dataset, performance improvement could, however, be made by tweaking the ConvNet architectural design using Adam and Nadam who were the better performance on this dataset.

3) Dataset 3 – Natural Images: Table 4 and Figures 19-27 presents the results obtained from natural images dataset, with Nadam the best performer in terms of validation accuracy and loss, closely followed by Adamax. And no significant difference between Adam (third) and SGDm (fourth), and the rest optimizer obtaining accuracies below 80%, except AdaGrad which had an accuracy of 69.9% and the worst performer on this dataset. It is also worth noting the improvement exhibited by SGDm (89.1%) in comparison to vSGD (77.5%), but performance dropping for SGDm+n (85.1%). Training was taken to converge at 140 epochs and an average of 5 hours convergence time across all the optimizers.

In terms of overall performance across all the three datasets, Nadam was the best performer attaining an accuracy rate of 85.5% on Cats and Dogs dataset, 91.3% on Natural images dataset and 71.2% on Fashion MNIST dataset, However, Nadam's performance was marginally lower than Adam (72%) its closest competitor on Fashion MNIST dataset. The experiments conducted confirms that performance of the optimizers is very much dependant on the type and size of dataset, with each optimizer exhibiting different level of accuracy and loss across the three datasets.

IV. CONCLUSION

In this paper, a comparative effect of seven optimization algorithms on three image classification datasets using a simple convolution architecture was carried out. Our findings reveal that the performance of each optimizer varied from each dataset, which confirms the effect that the data type and size plays on the performance of the different optimizers. Based on the several experiments conducted, our finding shows Nadam exhibited a more superior and robust performance across all the three datasets evaluated when compared to the other optimization techniques. This could be attributed to the fact that Nadam combines the strengths of Nesterov Acceleration Gradient (NAG) and Adaptive estimation (Adam) algorithms which apparently suits the three datasets and the model examined in this study.

In this study, only one model and three image classification datasets were used to perform all our experiments. It will be interesting to use more than three datasets of different problem domains and experiment on the comparative effects of these optimizers across a number of different ConvNet architectural designs and deep learning models for the purposes of better generalisation. We leave this for future endeavours.

ACKNOWLEDGEMENT

We would like to thank the University of Johannesburg for funding support as well as the valuable resources to complete this work.

REFERENCES

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal Computer Vision*, vol. 115, issue 3, pp. 211-252, 2015.

[2] ImageNet, "Large Scale Visual Recognition Challenge (ILSVRC)," vol. 2018, 2018.

[3] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. Retrieved from http://arxiv.org/abs/1609.04747, 29 October 2018

[4] K. Madsen and H.B. Nielsen, "Introduction to Optimization and Data Fitting," 2010.DTU Informatics - IMM.

[5] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, MIT Press, London, England, 2016.

[6] T. Schaul, I. Antonoglou and D. Silver, "Unit Tests for Stochastic Optimization," arXiv Preprint arXiv:1312.6055, Dec 20, 2013.

[7] G. Hinton, N. Srivastava and K. Swersky, "rmsprop: Divide the gradient by a running average of its recent magnitude," 2012.

[8] J. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121-2159, 2011.

[9] M.D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," arXiv Preprint arXiv:1212.5701, Dec 22, 2012.

[10] D.P. Kingma and L.J. Ba, "Adam: A Method for Stochastic Optimization," I*nternational Conference on Learning Representations (ICLR),* 2015.

[11] Sashank J. Reddi, Satyen Kale and Sanjiv Kumar, "ON THE CONVERGENCE OF ADAM AND BEYOND," *International Conference on Learning Representations (ICLR),* 2018.

[12] I. Sutskever, J. Martens, G. Dahl and G. Hinton, On the importance of initialization and momentum in deep learning, PMLR, Atlanta, Georgia, USA, pp. 1139-1147, 2013.

[13] K. Lv, S. Jiang and J. Li, "Learning Gradient Descent: Better Generalization and Longer Horizons," arxiv.org/abs/1703.03633, Mar 10, 2017.

[14] E. Yazan and M.F. Talu, "Comparison of the stochastic gradient descent based optimization techniques," *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pp. 1-5, Malatya, Turkey, 2017.

[15] G. Papamakarios, "Comparison of stochastic optimization algorithms," School of Mathematic, University of Edinburgh, 2014. Retrieved from https://www.maths.ed.ac.uk/~prichtar/papers/Papamakarios.pdf, 26 October 2014.

[16] Rasmus Hallen, "A Study of Gradient-Based Algorithms," 2017. Retrieved from http://lup.lub.lu.se/student-papers/record/8904399, 29 October 2018

[17] S. Shalev-Shwartz, O. Shamir and S. Shammah, "Failures of Gradient-Based Deep Learning," arXiv:1703.07950, 2017.

[18] J. Elson, J.R. Douceur, J. Howell and J.S. Asirra, "A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization," *Proceedings of the 14th ACM conference on computer and communications security (CCS)*, pp. 366-374, Oct 28, 2007.

[19] H. Xiao, K. Rasul and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," arXiv:1708.07747, 2017

[20] P. Roy, S. Ghosh, S. Bhattacharya and U. Pal, "Effects of Degradations on Deep Neural Network Architectures," arXiv:1807.10108, 2018.